Exhibit E

Knowledge Agents on the Web

Yariv Aridor[a] David Carmel[a] Ronny Lempel[b] Aya Soffer[a] Yoelle S. Maarek[a]

[a] IBM Research lab in Haifa, Matam, Haifa 31905, Israel
{yariv,carmel,ayas,yoelle}@il.ibm.com
[b] Computer Science Department, Technion, Haifa, Israel
rlempel@cs.technion.ac.il

Abstract

This paper introduces and evaluates a new paradigm, called K*nowledge Agents*, which incorporates agent technology towards domain-specific Web search in the context of dynamic domains. These domains are defined by the users and can be of any granularity and specialty. An agent stands between a user and a search engine and specializes in a specific domain by extracting useful information from search results. This information is saved in a private knowledge base to be used in future searches. Queries are refined by the agent based on its domain-specific knowledge and the refined queries are sent to general purpose search engines. The search results are ranked in the context of the agent's specific knowledge, thus filtering out pages which match the query but are irrelevant to the domain of interest. A topological search of the Web for additional relevant sites is conducted by the agent from a domain-specific perspective. The combination of a broad search of the entire Web with domain-specific textual and topological scoring of results, enables the knowledge agent to find the most relevant documents for a given query within the realm of the domain of interest. The knowledge acquired by agents is continuously updated and persistently stored so that users can benefit from search results of others in common domains of interest.

Keywords: Agent Technology; domain-specific Web search; Persistent Knowledge.

1. Introduction

The amount of information available on the World Wide Web is rapidly increasing on a daily basis. Existing general purpose search engines and browsers provide valuable assistance to users in locating the information relevant to their needs. However, finding information for a narrow query in a specific domain has become more and more difficult with the growth of the Web, and thus frequently resembles a search for a needle in a haystack.

Individuals spend more and more time filtering out irrelevant information returned from general purpose search engines while searching the Web. It is not uncommon for a user to obtain thousands of hits which match her query while belonging to different and irrelevant domains. The user then needs to traverse the list of retrieved documents in order to find the relevant ones. However, most users look only at one result screen (85% according to **[1]**), and are thus likely to miss many relevant pages. This is termed the *low precision problem,* as defined in information retrieval **[8]**.

There are couple of reasons for this low precision problem. First, users' queries are often short and not specific enough. In fact, statistics gathered from studying AltaVista's **[11]** log files indicate that the average number of terms in a query is 2.35 **[1]**. Furthermore, natural language ambiguity often results in users describing concepts in their queries in a different manner than authors describe the same concepts in their sites.

The low precision problem has long been recognized as a major difficulty in information retrieval systems for the Web. Many approaches have been suggested to tackle it **[7,12,18,22]**. One such approach is to restrict the search into a pre-defined domain. Several search services, most notably Yahoo! **[2]**, allow users to specify the domain in which to evaluate the query. Such restriction significantly reduces the number of irrelevant results and narrows the search space, hence increases the precision. However, the domain hierarchy is manually (or at best semi manually) crafted as a taxonomy of predefined categories. That is, the user has no control over which domains are included and cannot request a personal domain of interest. While this approach guarantees more quality, browsing is often time consuming, and of course coverage is extremely limited. Major search engines, such as AltaVista, index about two orders of magnitude more Web pages than Yahoo! **[3]**.

This paper presents a new paradigm, termed *Knowledge Agent*, which provides domain-specific search in the context of dynamic domains. With knowledge agents, domains are defined by the users themselves and can thus be of any granularity and specialty. In essence, it is an architecture which enables knowledge acquisition from search results which automatically characterizes the domain, in which the search was applied. This knowledge is persistently saved by the agent and can then be utilized to automatically narrow future searches within that domain. To the best of our knowledge no other systems utilizes knowledge acquired based on previous search results in future searches.

From a different perspective, this work originally explores the benefits of incorporating agent technology notions such as learning, autonomy, and personalization, into information retrieval technology. We vision knowledge agents benefits in other contexts than Web searching. Knowledge agents can assist in browsing, filtering, and routing information, to name a few possible applications, but this is outside the scope of this paper.

We have embodied this approach into initial prototype of a knowledge agent as a proof of concept of its benefits over general search engines. The major contributions we report in this work are:

- **The identification of the type of knowledge to be acquired from search results in order to characterize a domain of interest.**
- **A set of methods for knowledge acquisition and adaptation during the agent's operation.**
- **A system architecture which utilizes this knowledge for attaining high precision search results.**
- **A model of repository data persistent and easily transferable so users can benefit from search results of others with common domains of interest.**

The rest of the paper is organized as follows. Section **2** highlights the knowledge agent approach. Section **3** describes the system architecture in more detail. Section **4** provides some examples and experimental results. Section **5** discusses related work. Section **6** concludes the paper, highlighting future work and challenges.

2. Knowledge Agent Approach

The work described in this paper focuses on knowledge agents that assist users in searching the Web. We vision agents as autonomous software processes taking the role of expert librarians who assist in advanced search by using domain-specific knowledge, hence the term *knowledge agents*. Utilization of knowledge to prune the search space has been extensively studied in the artificial intelligence field [**4**,**24**]. Knowledge acquired by the agent can reduce future search efforts by directing the search into more desirable areas and by applying better analysis of the search results.

A knowledge agent acquires some special skills which are needed to provide quality search for material related to its own domain of expertise. Such domain specialization is characterized by:

- ***authority* pages - the most "definitive" pages pertaining to a specific domain, as defined by Kleinberg in [9].**
- ***hub* pages - pages containing leading references of information sources for this domain, as defined by Kleinberg in [9].**
- **a domain-specific terminology such as special abbreviations, special phrases, leading figures, etc.**

The knowledge agent (KA) maintains a knowledge base (KB) which consists of a set of leading sites of its domain (both hubs and authorities) and a repository of frequent terms that appear in these sites. Each term is associated with a list of *lexical affinities* **[5]** that we identify as closely related terms frequently found in proximity to that term. It has been described elsewhere how lexical affinities, when used as indexing units, improve precision of results, as compared to single words, especially in the context of domain-specific IR systems. The KB is updated continuously by the agent during search. New highly relevant pages found by the agent are entered into the KB, possibly taking the place of old pages with lower utility. Learning can also be accomplished by relevance feedback provided by the users. Pages manually marked as relevant to the domain will be entered by the agent into the KB. The KB can be initialized by providing a set of sites relevant to the domain of interest. This set could be extracted from the user's bookmark file or from any other existing pre-defined set of relevant Web sites.

From an architecture point of view, knowledge agents stand between the user and the search engine. The user's query is passed to the knowledge agent, which refines it and passes the refined query to the search process. The search process returns a set of results. The agent applies a ranking algorithm on this set, utilizing the knowledge of its domain of expertise. The ranked list of pages is returned to the user. Finally the agent updates its KB based on knowledge extracted from the results of the current search and feedback provided by the user. The rest of this section elaborates on these procedures.

Query refinement has long been recognized as an efficient tool for improving search results **[6]**. It is usually performed by adding terms related to the user's terms using a given thesaurus or a synonym table. Similarly, the knowledge agent refines the user's query using its domain-specific knowledge. The query is expanded by adding to each of the terms its most notable lexical affinities as found in the KB. The advantage of our approach is that the agent's local thesaurus characterizes the domain-specific ontology and thus relations between terms are domain dependent. As a result, added terms disambiguate the query terms in the context of the specific domain. For example, consider a search for the query "knowledge". An "artificial intelligence" agent would likely expand the query to include the terms "acquisition", "reasoning", "discovery", "representation", while a "cryptographic" agent would likely expand the query using the terms "zero", "private", etc.

Query refinement, however, is not sufficient for achieving high quality search results, especially when searching for a very specific query, since many relevant pages may not contain the exact terms that appear in the refined query. In order to improve the quality of the search results, the agent applies a topological search mechanism similar to the one applied by Clever **[7]**. It first compiles a list of candidate result pages by sending the refined query to one or several search engines. This basic set is then extended to include pages that are pointed to by pages in this set as well as pages that point to the pages in this set. Finally, the pages saved in the KB, which are assumed to be the most authoritative information sources for the given domain, are added to the retrieved set.

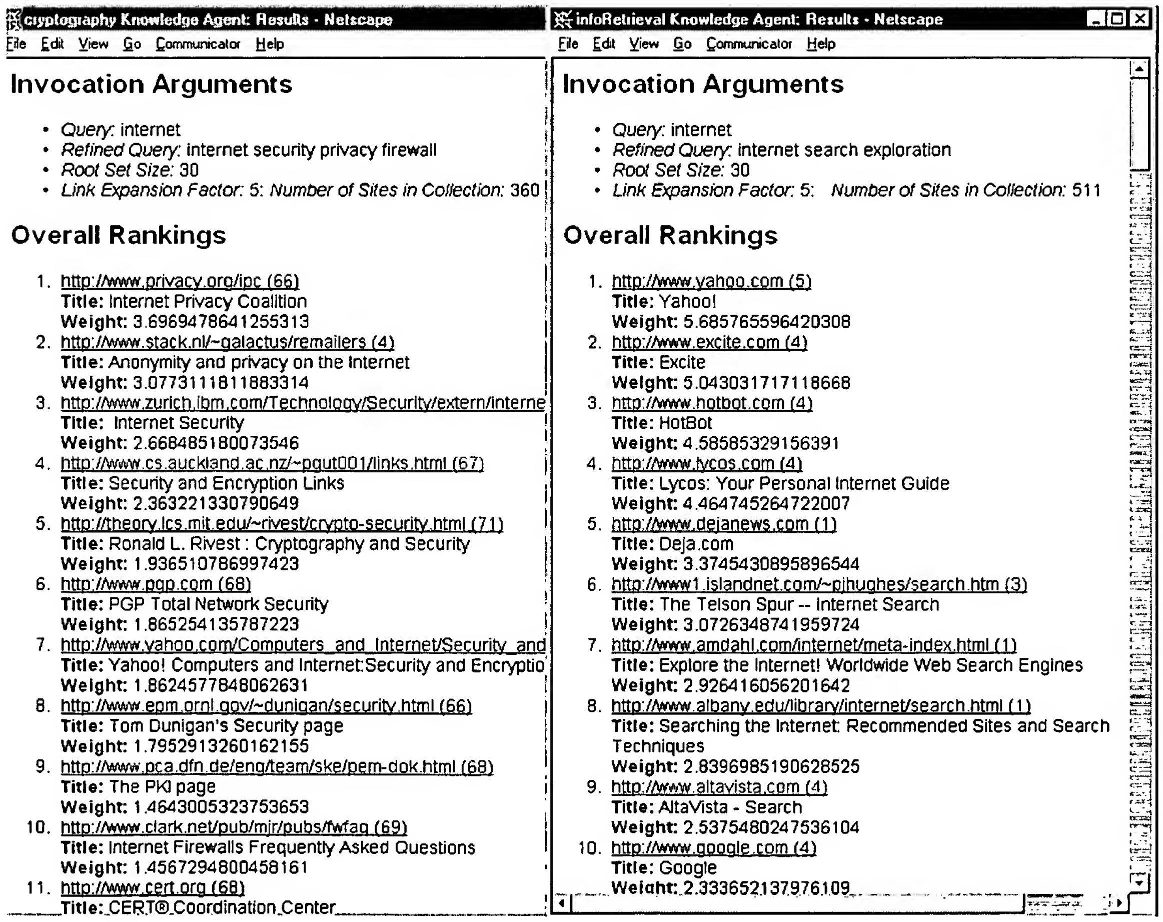
The expanded set of pages is then ranked by the agent, based on both textual and topological aspects, utilizing information stored in its KB. The textual similarity measures the relevance of the pages retrieved to the specific query as well as to the agent's domain. The similarity to the query as well as the similarity to the domain is measured using the well known tf-idf similarity measure of the vector space model [8]. The link topology score is a combination of an authority score and a hub score. The authority and hub scores are computed using a combination of Kleinberg s mutual reinforcement algorithm [9] as well as stochastic link analysis [10]. The overall score of a page is a linear combination of the textual similarity score and the topological link score. We elaborate on the ranking algorithm in section 3.2.

Computing the overall score based on both textual and topological aspects results in higher precision, however it incurs the overhead of response time. The knowledge agent analyzes the content of each page in the expanded set of pages, hence analysis time depends on the size of this set. The amount of expansion is a parameter of the KA system which is controlled by the user, thus users have control over the tradeoff between response time and quality of results. In general, low precision search results enforce users to later filter out irrelevant information manually. Knowledge agents strike a balance between the user efforts in analyzing low precision search results and time spent to find relevant information.

Taking advantage of the agent's knowledge in the search process is a key factor in providing high quality result. In particular, it forms the basis for providing:

1. **domain-specific query refinement.**
2. **domain-specific relevance evaluation of results: the KA performs a textual analysis of the returned pages in order to determine their relevance to the query in the context of a given domain whereas general purpose search engines base their rank only on the similarity to the query.**
3. **domain-specific results filtering: the KA filters out results that, while matching the query are not relevant to the domain by measuring the textual similarity of a page to the domain as a whole.**
4. **domain-specific hubs and authorities pages: the pages stored in the KB might often be good hits (or at least starting points) for queries in the agent's domain since they are leading sites for this domain.**
5. **domain-specific topological information: links between sites in the KB and result pages indicate relevance of these results to the domain.**

As a direct consequence knowledge agents that use a different KB will search the Web in a different manner and will produce different results. Each one retrieves a different set of relevant sites according to its personal point of view. Figure 1 shows the result of submitting the query "internet" to a cryptography (Crypto) agent and an information retrieval (IR) agent. The Crypto agent refined the query to include the terms "security privacy firewall", while the IR agent added the terms "search exploration". As the results show, each agent views the term "internet" in the context of its domain of expertise. The IR agent retrieved the main sites of the leading internet search engines, while the Crypto agent returned sites dealing with internet privacy and security.


cryptography Knowledge Agent: Results - Netscape
File Edit View Go Communicator Help

Invocation Arguments

- *Query:* internet
- *Refined Query:* internet security privacy firewall
- *Root Set Size:* 30
- *Link Expansion Factor:* 5: *Number of Sites in Collection:* 360

Overall Rankings

1. http://www.privacy.org/ipc (66)
   Title: Internet Privacy Coalition
   Weight: 3.6969478641255313
2. http://www.stack.nl/~galactus/remailers (4)
   Title: Anonymity and privacy on the Internet
   Weight: 3.0773111811883314
3. http://www.zurich.ibm.com/Technology/Security/extern/interne
   Title: Internet Security
   Weight: 2.668485180073546
4. http://www.cs.auckland.ac.nz/~pgut001/links.html (67)
   Title: Security and Encryption Links
   Weight: 2.363221330790649
5. http://theory.lcs.mit.edu/~rivest/crypto-security.html (71)
   Title: Ronald L. Rivest : Cryptography and Security
   Weight: 1.936510786997423
6. http://www.pgp.com (68)
   Title: PGP Total Network Security
   Weight: 1.865254135787223
7. http://www.yahoo.com/Computers_and_Internet/Security_and
   Title: Yahoo! Computers and Internet:Security and Encryptio
   Weight: 1.8624577848062631
8. http://www.epm.ornl.gov/~dunigan/security.html (66)
   Title: Tom Dunigan's Security page
   Weight: 1.7952913260162155
9. http://www.pca.dfn.de/eng/team/ske/pem-dok.html (68)
   Title: The PKI page
   Weight: 1.4643005323753653
10. http://www.clark.net/pub/mjr/pubs/fwfaq (69)
    Title: Internet Firewalls Frequently Asked Questions
    Weight: 1.4567294800458161
11. http://www.cert.org (68)
    Title: CERT® Coordination Center

infoRetrieval Knowledge Agent: Results - Netscape
File Edit View Go Communicator Help

Invocation Arguments

- *Query:* internet
- *Refined Query:* internet search exploration
- *Root Set Size:* 30
- *Link Expansion Factor:* 5: *Number of Sites in Collection:* 511

Overall Rankings

1. http://www.yahoo.com (5)
   Title: Yahoo!
   Weight: 5.685765596420308
2. http://www.excite.com (4)
   Title: Excite
   Weight: 5.043031717118668
3. http://www.hotbot.com (4)
   Title: HotBot
   Weight: 4.58585329156391
4. http://www.lycos.com (4)
   Title: Lycos: Your Personal Internet Guide
   Weight: 4.464745264722007
5. http://www.dejanews.com (1)
   Title: Deja.com
   Weight: 3.3745430895896544
6. http://www1.islandnet.com/~pjhughes/search.htm (3)
   Title: The Telson Spur -- Internet Search
   Weight: 3.0726348741959724
7. http://www.amdahl.com/internet/meta-index.html (1)
   Title: Explore the Internet! Worldwide Web Search Engines
   Weight: 2.926416056201642
8. http://www.albany.edu/library/internet/search.html (1)
   Title: Searching the Internet: Recommended Sites and Search Techniques
   Weight: 2.8396985190628525
9. http://www.altavista.com (4)
   Title: AltaVista - Search
   Weight: 2.5375480247536104
10. http://www.google.com (4)
    Title: Google
    Weight: 2.333652137976109

Figure 1: the results for the query "internet" using two different knowledge agents.

To summarize, the following are the main highlights of the knowledge agent approach:

- **Personalization: the user decides and maintains knowledge agents for his private domains of interests rather than being dependent on a fixed set of domains. These domains of interest can be of any granularity.**
- **Persistent knowledge: knowledge agents make it possible to utilize knowledge gained through search to improve search precision of future queries in the same domain.**
- **Global search: the knowledge agent searches the entire Web rather than a subset as in the case of domain-specific search engines.**
- **Portability: users can easily import knowledge agents created by others to assist in their search in any common domains of interests since the KB is implemented as a plug-in component of the knowledge agent software.**
- **Easy deployment: agents can be implemented as a front end to any search engine. It does not impose any extra functional overhead on the user compared to available search engines.**

3. System Architecture

The knowledge agent software prototype contains the following components:

1. **An Agent Manager, which sets up new knowledge agents and restarts existing agents. It is responsible for managing multiple concurrent connections for reading from the Web and serve multiple agents. The prototype currently performs meta-search by invoking three general purpose search engines - AltaVista [11], Google [12], Hotbot [13].**
2. **One or more Knowledge Agents which perform domain-specific searches in the Web. A detailed account of how these searches are conducted appears in section 3.2.**
3. **Each knowledge agent has an associated knowledge base which contains domain-specific information to be used for searching. The KB is updated continuously throughout the use of the agent. The structure of the information contained in the KB and the manner in which this information is updated is described in section 3.1.**

The general architecture of the system is illustrated in Figure 2.

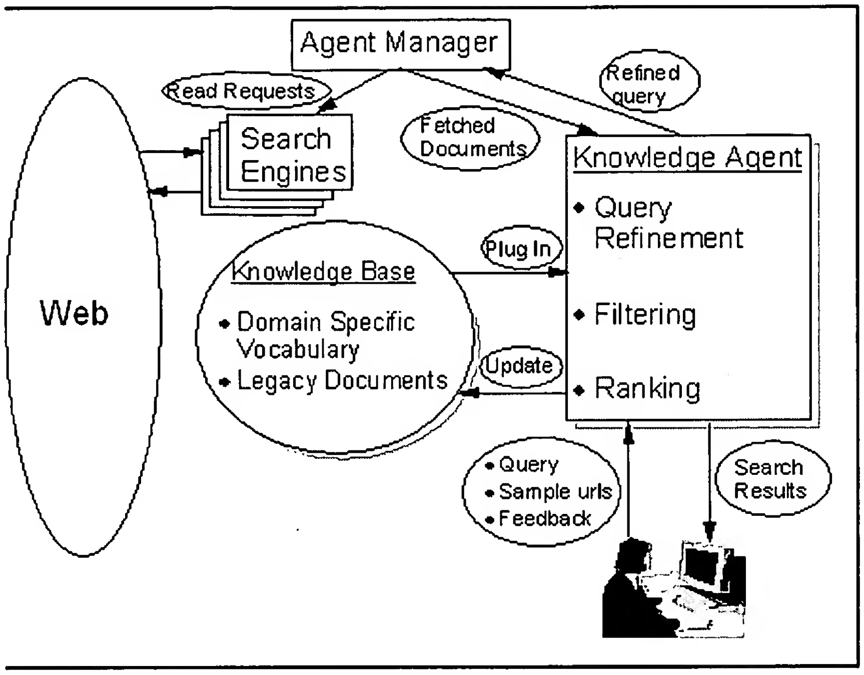


Figure 2: The knowledge agent architecture.

3.1 The Knowledge Base

The knowledge base contains a bounded collection of ranked sites and an aggregate profile of the textual content of these sites. Sites which are saved in the knowledge base are those which have proven to be highly relevant to a majority of the queries which users have submitted to the agent. The rationale for this is that sites which are consistently relevant to the users' queries are deemed as central to the domain in question.

The textual profile contains all of the words which appear in the sites, after deletion of stop words and a mild stemming process, along with their number of appearances. Each word *w* in the profile is associated with a list of its *lexical affinities i*dentified in the saved sites.

In order to enforce a limit on the number of sites in the KB, the flow of sites into and out of the KB should be regulated. The KB is adapted using an *evolutionary adaptation* mechanism. Sites fight for the right to be included in the agent's KB. Each kept site is assigned a *history score* which reflects the site's relevance to the domain through the course of the use of the agent. The combination of the history score and the relevance score for a specific query determines which sites are inserted into the KB and removed from it.

Sites can enter the KB in two ways:

- Through explicit insertion by the user. The user may supply a set of seed sites when a new KB is created. Such seeds may come from the user's bookmarks file, or from any other collection of sites which the user knows to be relevant to the domain in question. In addition, the user may add relevant sites to an existing KB at any point in time. If the KB contains the maximal number of sites allowed for it, the site with the lowest history score becomes stale and is removed from the KB, thus making room for the new fresh site. All sites which are entered into the KB explicitly by the user receive a high initial history score. This conveys our high regard for the user's judgment of the quality of these sites.
- Through automatic insertion by the agent. Upon completion of the *t*'th search process, the *t*-generation history score $h_t(s)$ of each KB site *s*, is updated in the following manner:

$$h_t(s) \leftarrow \beta_t \cdot h_{t-1}(s) + (1-\beta_t) \cdot S(s)$$

h_{t-1} is the history score of *s* prior to the *t*'th search,
β_t is a learning coefficient which controls the adaptation rate of the KB,
S(s) is the overall score of *s* for the *t*'th search (its computation is described in the next section).

The learning coefficient balances the two factors which set the value of the new history score of the KB□ s sites, namely the prior history score of the site, and its current specific score. The relative importance of the two components changes with the agent's *age*. As the number of queries performed by the agent grows, the weight of the history is increased. This reflects our confidence in KB sites of *mature* agents, which have processed many queries and are therefore more likely to be highly relevant to the domain in question. Thus, we set

$$\beta_t \leftarrow \beta_0 \cdot \delta^t$$

β_0 is an *initial* coefficient value.
δ is a decay factor which controls the rate of decay of the learning coefficient.

The new history scores of the KB sites are now compared against the overall scores of the new sites returned by the search. High score new sites may replace low score KB sites in the KB. Their initial history score is set to be their overall score for the current query.

The KB is a pluggable component of the KA. It can be saved to a file and restored from one, and thus knowledge can easily be transferred from one user to another.

3.2 The Search Process

The search process starts with the user entering a query and ends with the agent returning a ranked set of (hopefully highly relevant) sites to the user. Figure 3 describes the search process. The following sub-sections describe the associated procedures in more detail.

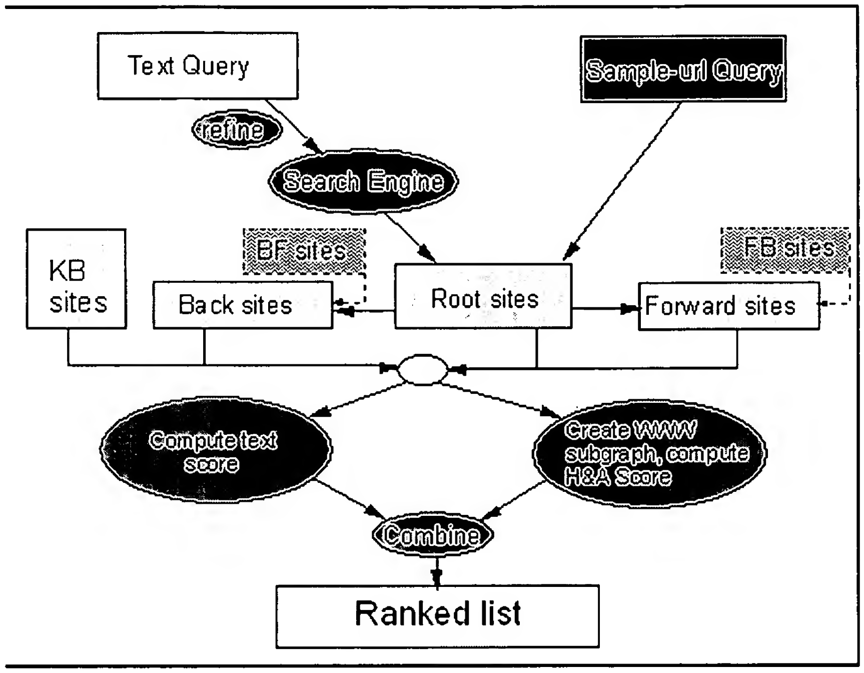


Figure 3: The search process applied by the agent.

3.2.1 Collecting the root set of sites

The system supports two kinds of queries, text queries and sample-url queries. A *text query* is a keyword based query such as those typically submitted to general purpose Web search engines. The user's query is automatically refined in the context of the agent's domain by adding to each of the keywords in the query its most notable lexical affinities as found in the profile of the KB. Since these added keywords occur most frequently in the vicinity of the original user's terms within the context of the agent's domain, they may improve the expressiveness of the query in several ways. First, the added keywords may disambiguate an ambiguous user query by supplying the underlying context of the domain to the query. Second, the added keywords can make the query more specific , thus aiding the search process in finding more precise results. Finally, the added keywords may be synonyms of keywords in the original query within the domain of interest, and thus relevant documents which would otherwise not be found, can be retrieved.

The Refinement factor, i.e. the number of terms which the agent adds to each of the user's original input terms, is controlled by the user. In addition, users can edit the refined query before submitting it to the search process. The refined query is then submitted to the user's choice of one or more general purpose search engines, and a user-specified number of results is collected from each of these engines. The set of returned results is called (following [9]) the *root set* of sites.

A *sample-url Query* is a query which specifies a few (typically *1-5*) seed urls, and whose purpose is to find a community of sites which are closely related to the seeds. Similar services are available by Excite's "More like this" feature [14] and by Google's "GoogleScout" feature [12]. However, both of these services receive as input a single site, while we allow the user to specify an arbitrary number of seeds. In sample-url queries, the user-supplied seed sites assume the role of the root set of sites, as if they were returned by a search engine in response to some textual query. The seed sites are then read, and their combined content serves as a pseudo query for the purpose of evaluating the textual content of other sites in the search process (as if the user had originally entered the entire text of the seed sites as the query).

3.2.2 Expanding the root sites

The collection of sites, which at the beginning contains just the root sites is expanded by following the hyperlinks surrounding the root sites, and by adding the sites which are stored in the KB (these are presumably central sites for the domain). The exact expansion model depends on the type of query which is being processed:

- **When processing a text query, the expansion follows the scheme presented in [9] and adds two sets of sites:**
  1. **The *Backward set B*, which contains sites which point to one or more root sites.**
  2. **The *Forward set F*, which contains sites which are pointed to by one or more root sites.**
- **When processing a sample-url query, the expansion phase must be more exhaustive and add more sites. This is because the size of the root set (the number of user-supplied seed urls) in such queries is much smaller than the size of the root set in text queries, and thus without a broader expansion phase, the resulting graphs would be too small and sparse for meaningful analysis. We thus follow ideas presented in [15], and add the following sets of sites to our collection:**
  1. **The previously mentioned sets, *B* and *F*.**
  2. **The set *BF*, which contains sites which point to one or more *F*-sites. Each of the *BF* sites thus shares an outgoing link with one of the seed sites.**
  3. **The set *FB*, which contains sites pointed to by one or more *B*-sites. Each of the *FB* sites thus shares an incoming link with one of the seed sites.**

In both expansion schemes, note that the url sets are not necessarily disjoint - a Web site may belong to more than one set.

The breadth of the expansion is controlled by the user, who specifies a *link expansion factor* for each query. This expansion factor, which is a natural number, specifies how many pointed/pointing sites will be added to the collection for each site in each expansion stage.
As noted previously, in both cases we also add to the collection all of the sites which are stored in the KB. We denote the entire collection of sites by ***C.***

3.3 Ranking the Web Pages

The agents receives a set of sites from the meta-search upon which it creates a ranked set of sites to be returned to the user. This section describes this ranking process.

3.3.1 Computing the textual score of a web page

With both query types, we have some text which defines the search topic, this being either a small possibly refined user provided query, or a larger pseudo query derived from the contents of a small set of seed sites. We create a profile consisting of each word in the text query (not including stop words) along with its lexical affinities, and compute a textual similarity score for each page with respect to the query profile. First we compute for each term in the query profile, both keyword and lexical affinity, a weight using a tf-idf formula originally used in the Guru search engine [**5**]. The textual profiles of the pages saved in the KB serve as the set of documents from which the terms' document frequencies are taken.

Some notations:

1. ***TC* denotes the total term count in the KB's profile (a term can be either a word or a lexical affinity)**
2. **Let *q(t)* denote the number of times the term *t* appears in the query *q*.**
3. **Let *kb(t)* denote the number of times the term *t* appears in the KB's profile.**

The weight of term *t*, *W(t)*, is defined as follows:

$$W(t) = q(t) * \log\left(\frac{TC}{\text{[illegible]}}\right)$$

The term weights are used to score the textual content of each site *s* with respect to a query using the following procedure. Text extracted from *s* is separated into three parts: strong, medium, regular. Strong text includes the words that appear in the title or in large font headers, medium text includes words that are either highlighted (bold, italics, etc.) or in small font headers, and regular text includes the rest. The textual score *T(s)* of site *s* is a weighted combination of the textual score for each text type:

$$T_q(s) = W_s \times \frac{\sum_{i=1}^{TC_s} w(t_i)}{TC_s} + W_m \times \frac{\sum_{i=1}^{TC_m} w(t_i)}{TC_m} + W_r \times \frac{\sum_{i=1}^{TC_r} w(t_i)}{TC_r}$$

TC_s, *TC_m*, *TC_r* denote the number of terms in the strong, medium, and regular text types, respectively. W_s, W_m, W_r, are constant weights assigned to each type $w(t_i)$of text that determine the influence of that text type in the total score. are the term weights computed by the term weight equation above for terms appearing in *the query profile*. They are set to zero for terms that appear in s but do not appear in the query profile. The similarity score for each text type *t* is thus the sum of the weights of each query term that appears in the web page in type *t*, normalized by the total number of terms of type *t* in the web page.

In addition we compute a textual similarity score $T_d(s)$ of each site to the domain. $T_d(s)$ is set to the dot product of the vectors of lexical affinities representing s and the domain. Term weights are assigned in relation to their frequency in the domain. The rationale for this is that pages that have many lexical affinities in common with the domain are most related to the domain. $T_q(s)$ and $T_d(s)$ are normalized and combined to create the overall textual similarity score *T(s)*.

3.3.2 Computing the link topology score of a web page

We build a Web subgraph, induced by the collection of sites ***C***, on which we perform connectivity analysis in order to find authoritative Web sites. The idea behind connectivity analysis is that a hyperlink from a site *s* to a site *t* indicates that these two sites share a common topic of interest, and that *s* conveys a positive assessment on *t*'s contents by recommending that surfers who visit *s* also visit *t*. We call such links *informative*.

In order to compute the link topology score, we first assign weights to the edges of the Web subgraph built during the search process. Every link receives a positive weight. The weight is set according to the anchor text associated with the link, and the "type" associated with the sites on both sides of the link (the source site and the target site of the directed hyperlink):

- ***Anchor Text contribution*: the anchor text is the method by which the pointing page describes the destination page to surfers, and is often a good source of information regarding the contents of the destination site. Thus, anchor text which resembles the query adds weight to the link which it describes (following [16,18]).**

- ***Anchor Links*: links which connect a KB site with a non-KB site (in either direction) are considered as more important, since they connect a site which is presumed to be central to the domain (the KB site) with a site which presumably answers the specific query. Such cross links are called *anchor links*, and their weight is increased by a constant additive factor (A similar idea was recently applied in [17]).**

This weighted Web subgraph is used to assign the hub and authority scores to each site. Each site *s* in ***C*** receives two hub and two authority scores from which a link topology score is derived.

- ***Mutual Reinforcement hub & authority scores, h1(s), a1(s)*. These are the hub & authority scores which the site receives when applying Kleinberg's Mutual Reinforcement algorithm [9] to the weighted graph at hand.**
- ***Stochastic hub & authority scores, h2(s), a2(s)*. These are the hub authority scores which the site receives by applying *SALSA*, the Stochastic Approach for Link Structure Analysis [10] to the weighted graph at hand.**

These scores are normalized and combined to form the *link topology score*, *L(s)*.

3.3.3 Computing the overall score of a Web page

The textual score and the link topology score are combined to yield the overall score of each site *s* in ***C***:

$$S(s) = f_C\, T(s) + (1-f_C)\, L(s)$$

Link topology scores are reliable only for collections in which many neighboring sites have been added around the meta-search results. We therefore set the value of f_C according to the ratio between the size of the compiled collection ***C*** and the size of the *root set*. The larger that ratio, the more confidence we have in the link-based score, and the lower we set f_C. When the ratio is low, meaning that the link expansion phase did not add many sites, we raise the influence of the text-based scores by raising f_C.

4. Examples and Experiments

We have developed a prototype system that implements the KA architecture as described in Section **3**. We created several agents using our system: a palm pilot agent, a cryptography agent (Crypto), an artificial intelligence agent (AI), a Geographic Information System agent (GIS), an Information Retrieval agent (IR), and a Star Wars agent. We trained each agent by submitting several textual queries relevant to its domain. Each agent's KB includes 50 to 100 urls (depending on the agent setup) as well as a textual profile representing the textual content of these pages. In this section, we present some examples and experimental results using these agents. The experiments were conducted as a proof of concept for the KA main functionality. Clearly, they by no means replace future exhaustive testing and evaluation study of the KA performance.

4.1 Query Refinement

We first examined the query refinement capabilities of the agents. We selected a few queries and examined the terms added by each KA while refining these queries. We made the following observations. The agents can complete names of people in their fields. For example, the Crypto agent adds "Ron" and "MIT" when asked about "Rivest", while the AI agent adds "Stuart", "Norvig", and "aima", when asked about "Russel" (aima is an an abbreviation for a famous introductory AI book written by Stuart Russel and Peter Norvig). They know about algorithms and theoretical aspects of their domains (e.g., the Crypto agent adds "Interactive" and "Proof" to "Zero Knowledge", adds both "Stream" and "Block" to the term "Ciphers"). They even know the marketplace (for the term "Checkpoint", the Crypto agent adds "firewall" and "vendor"). The agents are particularly good with acronyms. For example, the GIS agent expands "dlg" to "digital line graph data", "daac" to "distributed active archive center", and "eos" to "nasa earth observation system". For most of the larger agencies it lists what they deal with (usgs - digital map data, nasa - space center data earth, noaa - data satellite avhrr). It can also complete terms (e.g., for "latitude" it adds "longitude" and vice versa, for "coordinate" it adds "system"). For projects/instruments it gives their context (alexandria - digital library project, landsat - satellite nasa, hubble - telescope).

We then selected some common terms and examined how four of the agents refined them. Table 1 shows how different agents refine the same terms differently according to their domain-specific ontology. For example, the query "science" is expanded by the Crypto agent to include "computer" and "cryptography", by the GIS agent to include "earth" and "information", and by the Star Wars agent to include "fiction".

Agent*: *terms:	Palm Pilot	Cryptography	Geographic Information System (GIS)	Star Wars
software	development, download	encryption, security	esri (large software company)	game
knowledge		zero	opto systems	star war encyclopedia

public	license, gnu, domain	key, x509 certificate	domain, data	free site
science		cryptography, computer	earth, information	fiction
Microsoft	windows, operating system	crypto api	terraserver (ms geographic data server)	
unit	battery, modem	rsa ,certificate trusted	map boundary, remote sensing	hyperdrive
video	driver	stream	gis library remote sensing	clip picture sound

Table 1: query refinement performed by different agents.

4.2 Textual Analysis Examples

Our next example shows the agent's ability to sift up good results in the case that there are few hits returned by the meta-search. Assume that you are looking for a free French-English dictionary for your palm pilot on the Web. You will probably choose one of your favorite general purpose search engines, and submit a proper query such as: 'free "French-English dictionary" for palm pilot'. We submitted this query to AltaVista. It returned 579,478 pages. Many deal with French-English dictionaries, many others deal with palm pilots but not one of the top 25 pages provided a good answer. The 26th page was the first decent hit: a shareware French-English dictionary for palm pilots that can be downloaded for only $10.

We then submitted the same query to the palm pilot agent using AltaVista as the meta-search engine with a root set size of 50 and zero link expansion. We used a zero link expansion in order to neutralize the effects of the link topology score on the overall ranking. The palm agent re-ranked these same pages (i.e. the top 50 returned by AltaVista) by analyzing their relevance to the given query based on their textual similarity. The palm agent brought the perfect hit from the bottom of the AltaVista list up to the fifth place in the list. The first 4 pages in the agent's list deal with French-English dictionaries for palm pilots but are not perfect hits since they are not given for free. The agent's success is attributed to the textual analysis employed by the agent to the pages returned by AltaVista in the context of the palm pilot domain.

We then examined the overall precision of the agent's textual analysis without link expansion. We submitted some sample queries to the agents and compared the precision of the top results with the precision of general purpose search engines. Each query was submitted to AltaVista and Google, as well as to a knowledge agent specializing in the domain of the query. In one run The KA used AltaVista for meta-search and Google in the second run. The relevance of each site in the top results, was examined in the context of the domain of interest by an expert in the domain. Table 2 summarizes the results of these experiments.

		top@10				top@20			
Query	Agent	AltaVista	KA using AltaVista	Google	KA using Google	AltaVista	KA using AltaVista	Google	KA using Google
landsat 7 sample image	GIS	0.2	0.6	0.6	1	0.2	0.4	0.55	0.65
zero knowledge	Crypto	0.1	0.6	0.3	0.5	0.35	0.35	0.35	0.4
relevance feedback	IR	0.6	0.6	0.4	0.7	0.6	0.5	0.5	0.65

Table 2: precision at 10 and 20 using various search engines and a knowledge agent with zero link expansion.

From these results, it is apparent that performing textual analysis in the context of the domain of the specific KA improves the precision of the search significantly. The KA usually succeeds in sifting up relevant sites from the bottom of the meta-search results. For example, when submitting the query "landsat 7 sample image" to the GIS KA using AltaVista as the meta-search engine, the four most relevant sites, ranked 2, 5, 14, and 19 by AltaVista, were ranked 1 - 4 by the KA. In addition, two more relevant sites with numerous links to satellite images were returned in the top 10 hits. These sites were extracted from the agent's knowledge base. The top 10 sites returned by Google were related to landsat 7, however only six of these sites had sample images or links to such images. There were several more relevant sites in the next 20 hits. When submitting the same query to the GIS agent using Google as the meta-search engine, all ten sites in the top 10 where relevant. The four non-relevant sites in Google's top 10 were replaced by four relevant sites from further down the list.

Note that the improvement in the top@10 precision is more significant than the improvement in the top@20 precision since the non-relevant sites which were moved from the top of the meta-search results remain in the top 20.

It is important to note that the agent's success in these examples is attributed to the fact that the meaning of the query terms may differ according to the context of the domains. The KA was able to rank higher the sites more relevant to its in the particular domain of interest. It may be the case, that by expanding the query to include the term ▯ cryptography▯ to the query "zero knowledge", the general purpose search engines would be more successful. However, it has been shown in [18] that for queries containing both broad domain terms and narrow terms general search engines tend to return sites that are relevant to the broad domain and find it hard to retrieve sites dealing with the narrow topic within the context of the domain.

We tried to search for these same queries using Yahoo!. We traversed its hierarchy and found categories corresponding to our agent domains. Yahoo! was basically unable to return any valid hits for these queries. For the query "relevance feedback", Yahoo! found nothing relevant in the Information Retrieval category. In fact there were no hits when submitting this query to all of Yahoo!. Searching for "zero knowledge" under the cryptography category as well as the computer science category also resulted in zero hits. The query "landsat 7 sample image" also returned no hits in the GIS category. When searching all of Yahoo!, only three pages that are the main gateways to the Landsat 7 program were returned. They are placed in a category "Science > Space > Satellites > Missions > Landsat Program". None of them had any images. However, they contain links to pages were some images can be found. It seems that Yahoo! stores mostly high level authoritative sites for its categories and is thus unsuitable and not meant for searching for terms as specific as those in our examples.

4.3 Link Expansion

In order to test the effect of link expansion we executed the same queries with a link expansion of 5 (every site in the root set contributes 5 sites which it points to and 5 sites which point to it) using AltaVista for the meta-search. We compare these results to executing the same queries with zero link expansion. Table 3 summarizes the results.

Query	**Agent**	top@10		top@20	
		No Exp.	Exp.	No Exp.	Exp.

landsat 7 sample image	GIS	0.4	0.8	0.2	0.55
zero knowledge	Crypto	0.6	0.4	0.35	0.5
relevance feedback	IR ·	0.6	0.7	0.5	0.6

Table 3: precision at 10 and 20 using knowledge agents with zero link expansion and with link expansion 5.

In general, link expansion improved the results by finding additional relevant sites that were not returned by the meta-search but were either pointed to or pointed by these sites. For example, for the query "landsat 7 sample images", the GIS agent was able to retrieve several additional relevant sites compared to search with zero link expansion. There were 8 relevant sites in the top 10, ranked in places 1 - 8 (numbers 9 and 10 had sample images but not landsat images) compared to 4 with zero link expansion. Similar results were observed in the top 20. The improvement for the relevance feedback query was not as significant, since we already had god precision with zero link expansion. Nevertheless, we were able to add one more relevant site to the top 10 and two relevant sites to the top 20. Note that in the case of the query "zero knowledge" the precision at 10 decreased using link expansion since the agent retrieved two additional non-relevant sites and mistakenly ranked them in the top 10.

We tried to compare our results to those of Clever which also uses link expansion for retrieval. Clever was basically unable to find good sites for these queries. It returned some good IR hubs and authorities for the "relevance feedback" query, but was unable to locate even one site that contains this term. Similarly for the query "zero knowledge", it returned cryptography sites but non pertaining to zero knowledge. For the "landsat 7 sample image" query, it only returned the major landsat 7 gateways. These results are not surprising since Clever is not meant for these type of narrow queries [**17**].

While link expansion is a powerful tool for improving precision, its drawback is that time complexity increases with the size of the expansion. The KA system lets the user choose the desired link expansion, thus giving them control over the tradeoff between response time and quality of results.

5. Related Approaches

Many approaches have been suggested to tackle the low precision problem involved in searching the Web. Hierarchical categorization of Web sites is the one which most of the general purpose search engines support. It allows users to navigate through a hierarchy of categories and specify explicitly the domain in which to evaluate their query. Restricting the search to a specific domain reduces the number of irrelevant results, hence increases the precision significantly. However, as we already mentioned, Web categorization requires significant human effort and coverage is extremely limited.

Another alternative to restrict the search into a limited domain is by using a domain-specific search engine. Such engines are growing in popularity because they offer increased accuracy and extra functionality not possible with general search engines. They allow users to perform a focused search on a given topic within the scope of the specific domain covered by the search engine. For example, MRQE [19] allows users to search only for movie reviews. CampSearch [20] is another example which resembles database search. It allows complex queries over summer camps by age, size, location and cost. Performing any of these searches with a traditional general purpose search engine would be extremely tedious and most likely not yield to such accurate results. However, as in the case of manual categorization, building such search engines is a labor intensive process.

The search broker [21] is a meta-search tool which utilizes existing domain-specific search engines on the Web. Users specify the domain in addition to their query, and thus make it possible to apply a domain-specific search engine for their narrow query. Each search engine that the search broker is familiar with covers a certain domain, and each domain is associated with a list of terms (aliases) related to that domain. The search broker chooses the most proper search engine for the narrow query according to the domain specified by the user. The collection of search engines used by the search broker and the assignment of aliases that describe each engine are maintained by a human librarian.

A similar approach is a two phase search employed by Fetuccino-Alfredo [18]. There, users provide a broad domain in which the search should be performed in addition to their narrow query. Fetuccino-Alfredo first identifies sites related to the broad domain, using a general purpose search engine, and then dynamically searches for the narrow query by traversing the domain sites and their close neighbors. Fetuccino-Alfredo could benefit substantially from our knowledge agent approach by saving the domain maps for future usage, thus avoiding the recreation of these maps repeatedly.

Focused crawling [22] tries to automatically build a domain-specific search engine. The goal of a focused crawler is to selectively seek out pages that are relevant to a predefined domain. Rather than collecting and indexing all accessible Web documents to be able to answer all possible ad-hoc queries, a focused crawler finds the links that are likely to be most relevant for the specified domain, and thus avoids irrelevant regions of the Web. Jcentral [23] is such a search engine which allows Java developers to search for Java sources on the

Web. Jcentral crawls the entire Web but indexes only Java sources: Therefore, only pages that are related to the Java language are retrieved by the engine for any query.

The common feature of all of the aforementioned methods is focusing the search into a specific domain in order to improve search precision. The knowledge agents described in this work apply a similar strategy, however, they can automatically specialize in any domain as defined by their user and are therefore much more suitable for personal assistance. Furthermore, the knowledge acquired during the agent's activity is persistently kept by the agent and continually updated to enable improved search services in the future. Finally, the domain-specific textual profile maintained by the knowledge agent is a very powerful tool both for query refinement and for textual ranking of the relevancy of documents to a specific query within the domain.

6. Concluding Remarks

The abundance of information on the Web and its heterogeneous nature make it harder than ever to reach high precision results for a narrow query within a specific domain. Several search systems have tried to alleviate this problem by restricting search into a fixed set of predefined domains. Users have no control over the specific domains that are selected, the coverage of these systems is usually limited.

The knowledge agent approach presented in this paper suggests a new paradigm which provides domain-specific search in the context of dynamic domains. These domains are defined by the users and can thus be of any granularity and specialty. The agent's knowledge base is continuously updated using a learning process which extracts the most relevant information for the domain every time the knowledge agent is used. Queries are refined by the agent based on its domain-specific knowledge. The refined queries are sent to general purpose search engines and the results are ranked from the viewpoint of the specific knowledge agent, thus filtering out documents which match the query but are irrelevant to the domain of interest. Furthermore, a topological search of the web for additional relevant documents is also conducted from a domain-specific perspective. Topological search is an effective method for finding additional relevant sites for a query. The knowledge stored by the agent, enables it to enjoy the benefit of topological search, while traversing a relatively small search space.

The combination of a broad search of the entire Web, using general purpose search engines, with domain-specific textual and topological scoring of results, enables knowledge agents to find the most relevant documents at search time for a given query within the realm of the domain of interest. The knowledge acquired by knowledge agents is persistently stored, it is a pluggable component, and can be easily transferred to others. Knowledge can thus be shared within communities interested in the same domain, and members of the community can benefit from previous search efforts in this domain.

Knowledge agent is work in progress. We have implemented a prototype and conducted some initial experiments that show the potential benefits of this approach. There are however still several open questions. First and foremost is the question of how to best characterize a domain. Currently the domain is characterized by a list of leading sites and the entire textual content of these sites. Ideally, we would like to use only those terms that in fact distinguish this domain from others while filtering out irrelevant terms. Furthermore, sites found as most relevant to a particular query, might be entered automatically into the KB. While these sites should best characterize the domain, this is not always the case. A feedback mechanism whereby users could indicate the relevance of results to the domain as a whole can assist the learning process and improve the domain characterization. Finally, since the Web is constantly changing, letting the agent work autonomously off-line, looking for new sites similar to those already in its knowledge base, will be probably needed in order to keep the domain characterization up to-date. These are all subjects for future work.

In this paper we have described how knowledge agents can assist Web search. We vision knowledge agents as light components that can be plugged into several other Web applications such as browsing, filtering and routing systems. The knowledge acquired by the agent while searching for information pertaining to a user's domain of interest, can then assist in analyzing information acquired from other Internet sources from the point of view of this particular agent's domain.

Acknowledgments

We thank Dan Pelleg for helpful and useful discussions concerning the knowledge agent approach.

References

1. **Monika R. Henzinger and Krishna Bharat, Improved Algorithms for Topic Distillation in a Hyperlinked Environment, Proceedings of the 21'st International ACM SIGIR Conference on Research and Development in Information Retrieval, Aug. 1998**
2. **Yahoo!, http://www.yahoo.com**

3. Search Engine Watch, http://www.searchenginewatch.com
4. D. Michie, A Theory of Advice, Machine Intelligence 8, pages 151-170, 1977
5. Yoelle S. Maarek and F. Smadja, Full text indexing based on lexical relations, an application: Software libraries, Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 198-206, 1989.
6. Jinxi Xu and W. Bruce Croft, Query Expansion using Local and Global Document Analysis, In Proceedings of the 19th annual international ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pages 4-11.
7. IBM Research Center in Almaden, CLEVER, http://www.almaden.ibm.com/cs/k53/clever.html
8. G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, Computer Series, McGraw-Hill, New York, 1983
9. Jon M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, Proceedings of the ninth ACM-SIAM Symposium on Discrete Algorithms, 1998.
10. Ronny Lempel, Finding Authoritative Sites on the WWW (and Other Hyperlinked Media) by Analyzing the Web's Link-Structure}, Master thesis, Technion, Israel Institute of Technology, Haifa, Israel, 1999
11. Compaq Computer Corporation, AltaVista Net Guide, http://www.altavista.com/
12. Google Inc., Google Search Engine, http://www.google.com/
13. Wired Digital Inc., HotBot Search, http://www.hotbot.com/
14. Excite Inc., Excite Search, http://www.excite.com/
15. Ken Law, Thomas Tong and Alan Wong, Automatic Categorization based on Link Structure, http://www.stanford.edu/~tomtong/cs349/web.htm, 1999
16. Soumen Chakrabarti, Byron Dom, David Gibson, Jon M. Kleinberg, Prabhakar Raghavan and Sridhar Rajagopalan, Automatic resource list compilation by analyzing hyperlink structure and associated text, Proceedings of the seventh International WWW Conference, 1998.
17. Soumen Chakrabarti, Byron Dom, David Gibson, Jon M. Kleinberg, Prabhakar Raghavan, Sridhar Rajagopalan and A. Tomkins, Mining the Web's Link Structure, IEEE Computer, August, 1999.
18. I. Ben-Shaul, M. Herscovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim V. Soroka and S. Ur, Adding Support for Dynamic and Focused Search with Fetuccino, Proceedings of the Eighth International WWW Conference, Toronto Canada, 1999, pages 575-587.
19. Movie Review Query Engine, MRQE, http://www.mrqe.com
20. CampSearch, The Search Engine for Camps, http://www.campsearch.com
21. Udi Manber and Peter A. Bigot , The Search Broker, The First Usenix Symposium on Internet Technologies and Systems, Monterey CA, December 1997

22. Soumen Chakrabarti, Byron Dom, Martin ven den Berg, Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery, Proceedings of the Eighth International WWW Conference, Toronto Canada, 1999, pages 545 - 561.
23. IBM Jcentral, Search the Web for Java, http://www.jcentral.com/
24. A. Junghanns and J. Schaeffer, Search Versus Knowledge in Game-Playing Programs Revisited, Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan, pages 692--697, 1997.

Vitae

- **Yariv Aridor is a research staff member at the IBM Research Laboratory in Haifa, Israel. He received his MSc. and PhD. in Computer science from the Tel-Aviv university, Israel, in 1989 and 1995, respectively. His research interests include cluster architecture technologies, distributed and mobile object-oriented systems and agent technology. He has published over 10 papers in referred journals and conferences.**

David Carmel is a Research Staff Member at the IBM Research Laboratory in Haifa, Israel, and belongs to the 'Information Retrieval and Organization' Group. His research interests include information retrieval, multi-agent systems and artificial intelligence. He received his MsC and PhD in Computer science from the Technion, Israel institute of technology, in 1993 and 1997 respectively. David joined IBM in 1997 and has been involved with projects dealing with text mining, search applications, and information retrieval on the Web. He is currently involved in the Knowledge Agent project.

- **Ronny Lempel is a Ph.D. Student in the Department of Computer Science, Technion, Haifa, Israel, focussing on WWW link-structure analysis. He received his B.Sc. and M.Sc. from the same department in 1997 and 1999, respectively. During the Summer of 1999, Ronny joined the Knowledge Agent project in IBM's Haifa Research Laboratory.**
- **Aya Soffer is a Research Staff Member at the IBM Research Laboratory in Haifa, Israel, and belongs to the 'Information Retrieval and Organization' Group. Her research interests include information retrieval, pictorial information systems, and non-traditional database systems. She received her MS and PhD degrees in Computer science from the University of Maryland at College Park in 1992 and 1995, respectively.**
- **Yoelle S. Maarek is a Research Staff Member at the IBM Research Lab in Haifa, Israel and manages the "Information Retrieval and**

Organization" Group that counts about 15 members. Her research interests include information retrieval, Internet applications, and software reuse. She graduated from the "Ecole Nationale des Ponts et Chaussees", Paris, France, as well as received her D.E.A (graduate degree) in Computer Science from Paris VI University in 1985. She received a PhD from the Technion, Haifa, Israel, in January 1989. Before joining the Haifa Lab, Dr. Maarek was a research staff member at the IBM T.J. Watson Research Center for about 5 years. She serves on the program committees of several international conference and is a member of the Review Board of the WebNet Journal. She has published over 25 papers in referred journals and conferences.